Digital Communications by Satellite

J. J. SPILKER, JR. Ph.D.
Chairman, Stanford Telecommunications, Inc.

PRENTICE-HALL, INC., *Englewood Cliffs, New Jersey*

Library of Congress Cataloging in Publication Data

Spilker, J J

Digital communications by satellite.

(Prentice-Hall information and system sciences series)

Bibliography: p. 625

Includes index.

1. Artificial satellites in telecommunication.
2. Data transmission systems. I. Title.

TK5104.S64 621.38'0423 75–43878

ISBN 0–13–214155–8

10 9 8 7 6

Printed in the United States of America

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL OF CANADA, LTD., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*

It also should be noted that satellite communications involves a substantial time delay ($\geq$0.25 sec) and often rather high data rates ($\geq$100 Mbps). This combination can make Forward Error Correction (FEC) much more desirable than automatic request for retransmission (ARQ), because of the large costs of ARQ to store data at the transmitter until a verification signal is received or a request for repeat is received for a data block. In ARQ, blocks of data are transmitted with redundancy introduced for error detection. If a data block is received in error, the receiver sends the transmitter a request for retransmission. The use of FEC and ARQ together can be advantageous.

In this chapter we review the structure of convolutional codes, describe the structure of the Viterbi decoding algorithm, and discuss the error rate performance of the decoding algorithm for PSK and QPSK signals both with and without carrier reconstruction phase noise. Many of the results described in this chapter were first derived by Viterbi and his co-workers in the cited references.

15-2 CONVOLUTIONAL CODE STRUCTURE

A convolutional encoder with constraint length K is a K-stage shift register with n linear algebraic function generators, one for each output port. A rate 1/2 code produces two output bits for every input data bit. If one of these output bits is the original data bit, the code is called systematic. Figure 15-1 shows the structure of a simple nonsystematic rate 1/2 encoder of constraint length $K = 3$.

Assuming that the encoder starts in the all-zero state, the first four bits

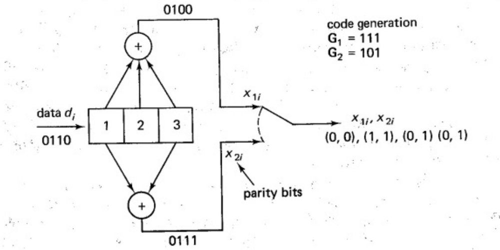

Fig. 15-1 Nonsystematic convolutional encoder with constraint length $K = 3$, and rate $1/n = 1/2$. The code generator denotes the tap positions.

0110 produce an output of 00, 11, 01, and 01, respectively, as shown. Clearly, the output of each new data bit depends on the previous bit pattern stored in stages 1, 2 of the shift register. These bit patterns can be labeled by the states defined as

$$a = 00 \qquad b = 01 \qquad c = 10 \qquad d = 11 \tag{15-1}$$

The output bits and transitions between states can be labeled by the trellis diagram of Fig. 15-2. The diagram starts in the all-zero state, node a, and makes transitions corresponding to the next data bit. These transitions are denoted by a solid line for a "0" and a dotted line for a "1." Thus, node a proceeds to node a or b with output bits 00 or 11.

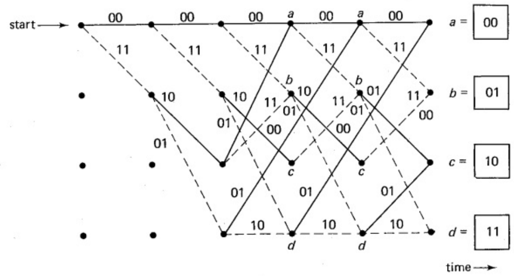

Fig. 15-2 Trellis-code representation for the convolutional encoder of Fig. 15-1

Table 15-1 shows the optimum codes for constraint lengths $K = 3$–8. The code generators define the taps for the K-bit shift register; n_e is the number of bit errors in paths at distance d_f, and d_f^* is the upper bound on minimum free distance [Odenwalder, 1970]. Notice that these codes are all nonsystematic. A systematic code would have $\mathbf{G}_1$ or $\mathbf{G}_2$ equal to 100 . . . 0; that is, one of the code generators would have only a single tap.

Note that some of the code structures in Table 15-1 are transparent to code inversion—that is, if the signs of the input bits are reversed, the coded output bit sequence is simply inverted. For example, if one of the parity bits x_{ji} is related to the data bits d_i by

$$x_{ji} = d_i \oplus d_{i-1} \oplus d_{i-2} \tag{15-2}$$

where there are an odd number of terms in the sum, reversing the sign of the d_i bits simply reverses all of these parity bits. Thus, if the numbers of "ones" or

Table 15-1 Optimum Rate 1/2 Codes (Maximum Minimum Distance) [Gilhousen et al., 1971]

Constraint Length K	*Code Generators*	*Distance d_f*	*Errors n_e*	*Distance Bound d_f^**	*Code Transparent to 180° Phase Reversal*
3	$G_1 = 111$ $G_2 = 101$	5	1	5	no
4	$G_1 = 1111$ $G_2 = 1101$	6	2	6	no
5	$G_1 = 11101$ $G_2 = 10011$	7	4	8	no
6	$G_1 = 111011$ $G_2 = 110001$	8	6	9	yes
7	$G_1 = 1111001$ $G_2 = 1011011$	10	36	10	yes
8	$G_1 = 11111001$ $G_2 = 10100111$	10	2	10	no

weights of both $\mathbf{G}_1$ and $\mathbf{G}_2$ are odd, then the code is transparent to a sign inversion. That is, the decoded output bit stream has the sign ambiguity as the input. This transparency is valuable if biphase-modulated PSK is used with its ensuing sign ambiguity for it permits decoding prior to ambiguity removal. Differential decoding at the decoder output removes the sign ambiguity and simply increases the output error rate by a factor of less than 2, because decoder output errors typically occur in short bursts. Differential decoding at the decoder input would double the decoder input error rate and thus would cause a much larger increase in the bit-error rate than a factor of 2 because of the high slope in the output-versus-input-error-rate curve.

15-3 THE MAXIMUM-LIKELIHOOD DECODER FOR A BINARY SYMMETRIC CHANNEL

Maximum-likelihood decoding could be accomplished over n coded two-bit symbols for rate 1/2 codes by comparing the received $2m$ output sequences with all $4(2^m)$ possible code paths leading to each of the 4 nodes in Fig. 15-2 and selecting the code sequences with the largest cross-correlation.* This calculation is extremely difficult for large m and would result in an overly complex decoder structure.

A major simplification was made by Viterbi in the likelihood calculation by noting that each of the 4 nodes has only two predecessors, and only the path with the highest cross-correlation weight need be retained for each node. For

*The factor of 4 includes all possible initial starting states.

Example survivor path to node *a*.

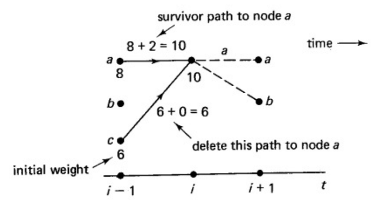

Fig. 15-3 Example of the iterative likelihood calculation showing the survivor path to node *a* at $t = i$. The alternate pattern is eliminated because of its lower weight.

example, the paths to node *a* might have weights 10 and 6 as shown in Fig. 15-3. At each node, the weight of the survivor path determines the new weight. For example, at $t = i$ the *aa* path might add a weight 2 to the weight 8 of the previous node *a*. The added weight can be computed by correlating the received code bits r_{1i}, r_{2i} with the parity bits for that transition p_{1i}, p_{2i} to produce $w_i = p_{1i}r_{1i} + p_{2i}r_{2i}$ where p, r are ± 1 for "hard" binary decisions in r_{ji}.

Figure 15-4 shows a typical path structure and weights for decoding. No errors have been introduced in the channel. Decoding has begun with no information as to the initial state (node) of the coder. Hence, all nodes at $t = 0$ have been set at zero weight. Notice that at $t = 5$ all node survivor paths began at node *a* at $t = 0$, the correct starting position. A decision on that $t = 0$ data bit, the *aa* path corresponding to a 0 data bit, can then be made. Thus in this example a correct data bit decision could be made with a 5 symbol delay. In general, with an error-producing channel, data bit decisions can be made after computing $5K$ successive nodes (a decoding delay of $5K$), where K is the coder constraint length.

If an error is made in selecting the data bit at any time instant, several data bits in succession may be decoded incorrectly before the correct path is reached. Figure 15-5 shows the surviving paths with an incorrect start position (start at node *d*) for the same data sequence as in Fig. 15-4. Note that in this example, 6 data bits, $t = 6$ (12 code bits), have been received before the correct path (correct path after $t = 2$) has produced the highest weight.

Some of the error correction and detection characteristics of the code can be established by redrawing the trellis in a state diagram (Fig. 15-6),

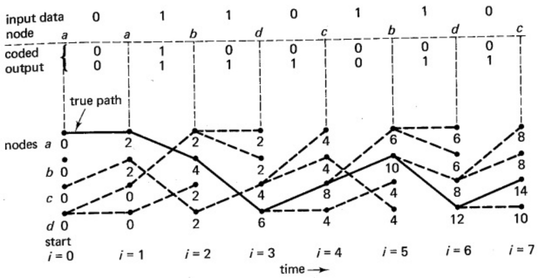

Fig. 15-4 Typical trellis pattern for decoded bit stream with code words received with no errors and unknown starting node. The true path is in solid line, and the other paths at each node are shown in dotted lines. The number adjacent to the node is its correlation metric. All nodes start at zero weight. Correlation is accomplished by replacing the codes by ±1 rather than 0, 1. Ties in the metric are broken by a random decision.

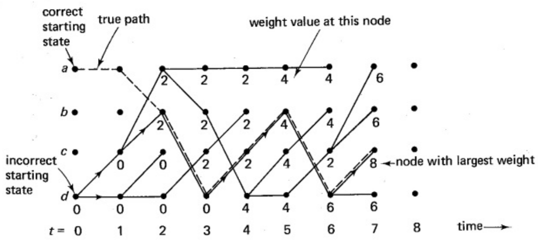

Fig. 15-5 Trellis paths after an initial starting error. The true path is shown by the dashed line.

where each path in the state diagram is labeled with the code bits corresponding to that path—for example, the *aa* path is 00. Note that the minimum-weight path from *a* to *a*, other than the direct *aa* path, is path *abca* shown by the dotted line. If *aaaa* is the correct path in three steps, the minimum alter-

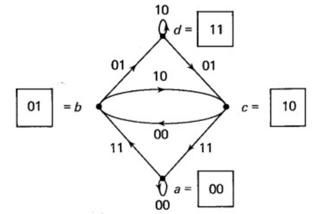

Fig. 15-6 State-diagram representation for coder of Fig. 15-4

nate-weight path *abca* corresponds to 11, 10, 11, and has weight 5, corresponding to five errors or Hamming distance $d = 5$. Thus, this code corrects any two errors over that path length and detects any three errors. Codes of minimum distance $d = 2e + 1$ can be used to correct e errors.

The error-correction properties of any code can be determined by generating the flow diagram from a to a for all paths, each labeled D^k, where k is the weight in that path. The flow diagram for the example code is given in Fig. 15-7. This diagram can be reduced to the generating function for all paths which eventually merge with the all-zeros path by the following calculations of the paths leading to each of the four nodes:

$$d = Dd + Db \quad \text{or} \quad d = \frac{D}{1-D}b \tag{15-3}$$

$$c = Dd + Db - b \quad \text{or} \quad c = \left[\frac{D}{1-D} - 1\right]b \tag{15-4}$$

$$a' = D^2c \quad \text{or} \quad a' = D^2\left[\frac{D}{1-D} - 1\right]b \tag{15-5}$$

$$\text{and thus } b = D^2a + c - Dd - Dc \tag{15-6}$$

Solving for a' in terms of a, we obtain from (15-3) to (15-6)

$$T(D) = \frac{a'}{a} = \frac{D^5}{1-2D} = D^5 + 2D^6 + 4D^7 + \cdots + 2^kD^{k+5} + \cdots \tag{15-7}$$

Thus, there is one path of weight 5, two of weight 6, and, in general, 2^k paths of weight $k + 5$.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.